VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Aplikace pro offline navedení uživatele do oblasti se signálem mobilní sítě

Android-based Offline Mobile Network Signal Finder

2013 Miroslav Sedlák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Miroslav Sedlák**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Aplikace pro offline navedení uživatele do oblasti se signálem mobilní sítě
Android-based Offline Mobile Network Signal Finder

Zásady pro vypracování:

I přes stále se zlepšující pokrytí signálem mobilních operátorů jsou v ČR (zejména v horských oblastech) místa, kde signál některého z operátorů chybí. Cílem práce je tvorba aplikace pro platformu Android, která bude umožňovat uživateli na základě aktuální polohy určit vhodná místa, kde by měl být signál operátora dostupný. Využita bude off-line databáze BTS, uložená na zařízení, výšková data SRTM, mapová komponenta a popř. mapová vrstva pokrytí operátorů signálem.

1. Zjistěte, jaké možnosti lokalizace poskytuje platforma Android, jaké informace o BTS a pokrytí jejich signálem jsou k dispozici z veřejných zdrojů (a v jaké podobě) a zda existují aplikace s podobným či stejným zaměřením.
2. Určete, jakým způsobem budou informace uloženy, sdíleny či získávány klienty (HTTP, cloudové řešení, webové služby, apod.) a jak bude zajištěna jejich dostupnost v off-line režimu. Zdokumentujte formát, v němž budou informace předávány. Vyberte, které části Android API využijete a v jaké verzi. Popište případné nestandardní knihovny a nástroje, které budete využívat.
3. Analyzujte, navrhněte a implementujte mobilní aplikaci (a případnou serverovou část).
4. Výsledné řešení otestujte alespoň na dvou různých mobilních zařízeních a zhodnoťte dosažené výsledky. Srovnejte funkčnost Vaší aplikace s aplikacemi z bodu 1.

Seznam doporučené odborné literatury:

[1] Burnette, E. Hello, Android: Introducing Google's Mobile Development Platform. Pragmatic Bookshelf, 2008. ISBN: 978-1-93435-617-3.
[2] Meier, R. Professional Android 2 Application Development. Wrox Press, 2010. ISBN: 0-47056-552-0.
[3] Android Developers [online][cit. 2012-10-01]. Dostupné z: http://developer.android.com/

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Moravec, Ph.D.**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave 7. 5. 2013 Sedlák .....

Týmto sa chcem poďakovať predovšetkým vedúcemu mojej práce Ing. Pavlovi Moravcovi, Ph.D. za obrovskú trpezlivosť a ochotu pri vedení práce. Tiež chcem poďakovať svojej rodine a blízkym za neustálu podporu.

Abstrakt

Cieľom bakalárskej práce je vývoj aplikácie pre mobilnú platformu Android, ktorá užívateľa navedie k miestu s dostupným pokrytím signálom operátora. Táto práca popisuje analýzu, návrh a implementáciu vytváranej aplikácie.

Kľúčové slová: Android, mobilná aplikácia, Java, SRTM, BTS, hľadanie signálu, offline, mapa pokrytia

Abstract

The goal of this bachelor thesis is to develop an application for the Android mobile platform, which will provide user with option to reach the place with cellular network coverage. This thesis describes analysis, design and implementation of the created application.

Keywords: Android, mobile application, Java, SRTM, BTS, signal finder, offline, coverage map

Zoznam použitých skratiek a symbolov

API	–	Application Programming Interface
BTS	–	Base Transceiver Station
CSV	–	Comma Separated Values
DAO	–	Data Access Object
DB	–	Databáza
GPS	–	Global Positioning System
HTML	–	Hyper Text Markup Language
LIFO	–	Last In First Out
NASA	–	National Aeronautics and Space Administration
NGA	–	National Geospatial–Intelligence Agency
NIMA	–	National Imagery and Mapping Agency
OS	–	Operačný Systém
RSSI	–	Received Signal Strength Indication
SDK	–	Software Development Kit
SRTM	–	Shuttle Radar Topography Mission
SQL	–	Structured Query Language
TXT	–	Textový formát súboru
UI	–	User Interface – užívateľské rozhranie
XML	–	eXtensible Markup Language

Obsah

Zoznam tabuliek

Zoznam obrázkov

Zoznam výpisov zdrojového kódu

1 Úvod

V Českej republike sa ešte stále nachádzajú miesta, najmä v horských oblastiach, kde signál mobilných operátorov nie je dostupný. Táto aplikácia slúži ako navigácia k najbližšej oblasti pokrytej signálom mobilného operátora. Je určená predovšetkým pre ľudí, ktorí majú v obľube turistiku, ale aj pre bežných ľudí, ktorí sa ocitnú v situácii, kedy na mobilnom zariadení nemajú dostupný signál operátora a potrebujú vedieť, kde je najbližšie miesto s pokrytím.

V predkladanej bakalárskej práci sa zameriavam na získavanie BTS dát a spôsob ich uloženia v mobilnom zariadení. Okrem týchto dát využívam tiež mapy pokrytia operátorov.

Impulzom pre výber bakalárskej práce bol záujem o túto problematiku a takisto snaha o doplnenie už existujúcich aplikácií s podobným zameraním pre platformu Android, ktoré neposkytujú riešenie danej problematiky bez aktívneho internetového pripojenia.

Podstatou a cieľom práce je vytvorenie novej aplikácie pre operačný systém Android, ktorá užívateľovi poskytne možnosť určiť vhodné miesta s dostupným signálom operátora na základe jeho aktuálnej polohy. Pri vytváraní aplikácie som pracoval s údajmi o BTS na vytvorenie offline databázy, výškovými dátami SRTM a mapovou vrstvou pokrytia signálom operátorov, ktoré som do nej integrovali.

Celkovo je práca členená do 8 kapitol. V druhej kapitole sú v krátkosti načrtnuté teoretické poznatky o systéme Android, ktoré nám pomôžu pri orientovaní sa v tejto problematike. Tretia kapitola poskytuje základné informácie o SRTM a dátach ňou získaných. Štvrtá kapitola sa zaoberá rozborom a analýzou samotnej aplikácie. Zaoberám sa v nej komparáciou s inými aplikáciami príbuzného charakteru a spôsobom získavania dát. Ďalej sú diagramy a dátový slovník, potrebné pre implementáciu. Návrh aplikácie je zahrnutý v piatej kapitole. Obsahuje návrh užívateľského rozhrania a diagram nasadenia. Po získaní poznatkov, zozbieraní údajov a vytvorení návrhu som pristúpil k najpodstatnejšej časti, implementácii a vytvoreniu aplikácie, o čom pojednáva šiesta kapitola. Problémy, s ktorými som sa stretol pri vytváraní aplikácie, a ich riešenia popisujem v tretej časti tejto kapitoly. V siedmej kapitole je popísaná požadovaná funkčnosť aplikácie a jej zhodnotenie. Na záver v ôsmej kapitole popíšem dosiahnuté výsledky a vyhliadky do budúcna.

2 Android

Android je open-source platforma, ktorá spája OS, middleware, mobilné aplikácie a množinu knižníc API [1]. Architektúra platformy Android je založená na komponentách. Android poskytuje množstvo vstavaných služieb ako napríklad služba na určovanie polohy alebo ukladanie dát do SQL databázy [2].

Komponentami aplikácie v systéme Android sú:

- Aktivity (Activities)
- Služby (Services)
- Content providers
- Broadcast receivers

Detailný popis týchto komponent nasleduje v ďalších podkapitolách. Informácie v nich vychádzajú z [4].

2.1 Android Manifest

Každá aplikácia obsahuje súbor *AndroidManifest.xml*. Ten sa musí nachádzať v hlavnom adresári aplikácie. Obsahuje informácie o aplikácii a jej komponentách, ktoré musí systém vedieť skôr ako ju spustí. Pomenováva Java balík aplikácie. Názov balíka slúži na identifikáciu aplikácie. V súbore *AndroidManifest.xml* sú deklarované komponenty, z ktorých sa aplikácia skladá. Určuje, v ktorom procese pobežia komponenty aplikácie, aké oprávnenia aplikácia vyžaduje pre prístup ku chráneným častiam API, alebo interakciu s inými aplikáciami a naopak, aké oprávnenia musia mať aplikácie, ktoré s ňou chcú spolupracovať. Deklaruje minimálnu, cieľovú a maximálnu úroveň Android API, ktorú aplikácia potrebuje, pričom povinná je len minimálna úroveň. Pokiaľ aplikácia používa knižnice, ktoré nie sú automaticky pripojené (napr. Google Maps library), musia byť deklarované pomocou <uses-library> [3].

2.2 Google Maps

Google Maps Android API nám umožňuje vytvárať aplikácie, ktoré využívajú dáta z Google Maps. Aktuálna verzia Google Maps Android API je verzia 2. API sa stará o prístup na servery, sťahovanie dát, zobrazenie mapy a reakcie na gestá, použité na mape. API nám poskytuje tiež možnosť pridať grafické objekty, ktoré môžu užívateľovi poskytnúť ďalšie informácie podľa potreby aplikácie. Tieto objekty môžu byť:

- ikony s pevnou pozíciou na mape (Markers)
- možiny úsečiek (Polylines)
- uzatvorené úseky (Polygons)
- Bitmapy s pevnou pozíciou na mape (Ground Overlays)

- množiny obrázkov, ktoré sú zobrazené na dlaždiciach základnej mapy (Tile Overlays)

Na vytvorenie aplikácie využívajúcej Google Maps Android API v2 potrebujeme SDK služieb Google Play, získať API kľúč, pridať potrebné nastavenia a oprávnenia do *AndroidManifest.xml* (viď Výpis 1) a pridať mapu do aplikácie (viď Výpis 2 a Výpis 3) [12].

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.mapdemo"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="7"
        android:targetSdkVersion="17" />

    <permission
        android:name="com.example.mapdemo.permission.MAPS_RECEIVE"
        android:protectionLevel="signature"/>
    <uses-permission android:name="com.example.mapdemo.permission.MAPS_RECEIVE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="com.google.android.providers.gsf.permission.
        READ_GSERVICES"/>
    <!-- The following two permissions are not required to use Google Maps Android API v2, but
        are recommended. -->
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>

    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true"/>

    <application
    .
    .
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="your_api_key"/>
    </application>
</manifest>
```

Výpis 1: AndroidManifest.xml [12]

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:name="com.google.android.gms.maps.MapFragment"/>
```

Výpis 2: Main.xml [12]

```
package com.example.mapdemo;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Výpis 3: MainActivity.java [12]

2.3 Activity

Aktivita (*Activity*) je komponenta aplikácie, ktorá poskytuje užívateľovi obrazovku, s ktorou môže interagovať. Každá aktivita má pridelené okno, v ktorom vykreslí svoje UI.

Aplikácia môže obsahovať viacero aktivít. Napríklad aktivita na zobrazenie počasia, ďalšia na pridávanie a odoberanie zobrazovaných miest a ďalšia s nastaveniami. Aktivity na seba nadväzujú. Aplikácia má zvyčajne jednu hlavnú aktivitu, ktorá je pri spustení zobrazená užívateľovi ako prvá. Každá aktivita môže v prípade potreby spustiť inú *Activity* (napríklad nastavenia aplikácie). V takom prípade sa predošlá aktivita zastaví, a systém ju uchováva v zásobníku (”back stack”). Tento zásobník je typu LIFO. Pri spustení novej aktivity sa uloží do zásobníka. Keď užívateľ danú aktivitu ukončí, je vyhodená zo zásobníka a do popredia sa vracia predchádzajúca aktivita.

Keď je aktivita zastavená kvôli spusteniu inej aktivity, je o tejto zmene informovaná pomocou metód spätných volaní životného cyklu aktivity. Aktivita môže byť v stavoch, kedy ju systém vytvára, zastavuje, vracia do popredia, alebo ruší. Podľa stavu, v akom sa daná aktivita nachádza, môžu byť vykonané potrebné operácie zodpovedajúce tomuto stavu. Napríklad pri zastavení aktivity je dobrým zvykom odstrániť z pamäte veľké objekty, ktoré by v nej zaberali zbytočne priestor [13].

2.4 Service

Služba (Service) na rozdiel od aktivity užívateľovi neposkytuje grafické rozhranie. Táto komponenta slúži na vykonávanie operácií v pozadí. Vďaka tomu nezaťažuje hlavné vlákno a nijak neruší užívateľa pri práci s aplikáciou. Užívateľ môže s aplikáciou pracovať, zatiaľ čo v pozadí služba vykonáva dlhé operácie (sťahovanie dát, výpočty, prehrávanie hudby). Službu môže spustiť iná komponenta, ako napríklad aktivita.

2.5 Content provider

Cez *ContentProvider* aplikácia umožňuje prístup k svojim dátam iným aplikáciám. Je jedno, kde sú dáta uložené (súbor, lokálna databáza, databáza na serveri, web). Pokiaľ má k nim aplikácia prístup, môže ho poskytnúť ostatným aplikáciám cez *ContentProvider*. Systém Android napríklad poskytuje aplikáciám možnosť čítať, alebo zapisovať kontakty užívateľa. Na to používa práve *ContentProvider*, ktorý spravuje informácie o kontaktoch užívateľa.

2.6 Broadcast receiver

Broadcast receiver je komponenta, ktorá zachytáva správy od systému a odpovedá na ne. Takéto správy sú napríklad vypnutie displeja, nízky stav batérie, alebo zachytenie snímky. Aplikácie môžu vytvárať vlastné správy, ktoré sú potom zachytávane pomocou *BroadcastReceiver*. Môžu nimi upozorniť ostatné aplikácie napríklad na ukončenie práce s dátami, ktoré im poskytujú. *BroadcastReceiver* síce neposkytuje žiadne UI, ale môže vytvárať notifikácie v stavovom riadku zariadenia, ktorými môžu užívateľa upozorniť na udalosť. V praxi sa *BroadcastReceiver* používa ako brána k ďalším komponentám. Napríklad môže spustiť službu, ktorá na základe obdržanej správy vykoná nejakú prácu.

2.7 AsyncTask

Abstraktná trieda *AsnyncTask* je určená na vykonávanie náročných úloh v ďalšom vlákne. Je vhodné ju použiť, ak chceme vykonať na pozadí operáciu, ktorá by zaťažovala hlavné vlákno, ale výsledky operácie chceme nejakým spôsobom zobraziť v UI (pracovať s nimi v hlavnom vlákne). Táto operácia by mala trvať najviac niekoľko sekúnd. Ak chceme, aby vlákno bežalo dlhšie, nemali by sme používať triedu *AsyncTask*, ale niektoré triedy z API *java.uti.concurrent*.

Operácie, ktoré by zaťažovali hlavné vlákno, je potrebné vykonávať v metóde *doInBackground(Params..)*. Naopak všetky operácie, ktoré potrebujeme vykonávať v hlavnom vlákne (napr. zobrazenie stiahnutých obrázkov, zobrazenie dialógu s priebehom), musia byť v jednej z metód *onPreExecute()*, *onProgressUpdate(Progres..)*, alebo *onPostExecute(Result..)* [5].

Pre použitie triedy *AsyncTask* je potrebné ju rozšíriť a prepísať jej metódu *doInBackround* a v prípade potreby aj metódy vykonávané v hlavnom vlákne.

3 Shuttle Radar Topography Mission

3.1 Čo je SRTM?

Jedná sa o misiu NASA, uskutočnenú vo februári roku 2000, počas ktorej boli získavané výškové dáta Zeme. SRTM tvoril špeciálne upravený radarový systém, ktorý letel na palube raketoplánu Endeavour počas 11-dennej misie. SRTM vznikla na základe spolupráce NASA a NGA (vtedy NIMA) [14].

V SRTM bola použitá technika nazývaná radarová interferometria. Táto technika spočíva vo vytvorení dvoch radarových snímok z mierne odlišných polôh. Rozdiely medzi týmito snímkami umožňujú výpočet nadmorskej výšky povrchu. Aby bolo možné zachytiť dve snímky z rozličných polôh, hardware SRTM pozostával z jedného radaru umiestneného v nákladnom priestore raketoplánu a druhého radaru pripevneného na koniec ramena, ktoré bolo 60 metrov od raketoplánu [15].

3.2 Verzie SRTM

Boli vydané dve verzie SRTM dát. Verzia 1 je vhodná na vedecké účely. Jedná sa o originálne dáta získané SRTM v roku 2000. Tieto dáta nie sú nijak upravované a obsahujú chyby.

Verzia 2 je výsledkom editovania NGA. Dáta stále obsahujú možstvo malých medzier a zopár väčších dier.

Verzia 2.1 je prepočtom verzie SRTM3. Vznikla 3x3 priemerovaním dát s plným rozlíšením (SRTM1) [16].

Existujú tiež verzie SRTM dát, ktoré neobsahujú chyby. V týchto verziách boli diery vyplnené použitím interpolačných algoritmov. V pôvodných dátach je dokopy 3436585 dier, ktoré predstavujú $796217km^2$. Častými úsekmi s chýbajúcimi dátami sú oblasti hôr, alebo púšte. Ďalšie informácie a popis algoritmu na vyplnenie dier v SRTM dátach je možné nájsť v [17].

Nasledujúce informácie vychádzajú z dokumentácie SRTM ([18]). SRTM dáta sú uložené v súboroch s príponou .HGT a ich názvy zodpovedajú zemepisnej šírke a dĺžke ľavého spodného rohu dlaždice. Teda sú vo formáte "N48E016", kde N predstavuje severnú šírku a E východnú dĺžku. Čísla predstavujú súradnice dlaždice v stupňoch.

Hodnoty v týchto súboroch sú typu short a sú uložené v poradí "big-endian". Pre prácu s týmito dátami je teda potrebné prehodiť poradie bajtov, pretože systém Android používa poradie "little-endian". Verzia SRTM3 obsahuje 1201 riadkov a v každom riadku 1201 hodnôt. Riadky na severnom a južnom okraji sa vždy prekrývajú s riadkami priľahlých buniek. To isté platí pre stĺpce na východnom a západnom okraji. Verzia SRTM1 obsahuje až 3601 riadkov, v ktorých je 3601 hodnôt. Riadky a stĺpce sa prekrývajú tak ako je to pri SRTM3. Tieto dáta sú detailnejšie (vzorkovanie 1 uhlová sekunda). Diery v dátach predstavuje hodnota -32768.

3.3 Použitie v aplikácii

V našej aplikácii je možné použiť jednu z verzií SRTM3 dát. Výber verzie je na užívateľovi. V prípade výberu verzie 1 treba rátať s jej chybami. Pri testovaní boli použité SRTM3 dáta vo verzii 2.1, ktoré sú dostupné na `http://dds.cr.usgs.gov/srtm/version2_1/SRTM3/Eurasia/`.

4 Analýza

Táto kapitola sa zaoberá analýzou vytváranej aplikácie. V popise zhrnieme očakávané funkcie aplikácie. Potom porovnáme súčasné riešenia aplikácií, ktoré nejakým spôsobom poskytujú informácie o pokrytí signálom. Toto porovnanie zhrnieme do tabuľky. Nakoniec vykonáme dátovú a funčnú analýzu.

4.1 Popis aplikácie

SigFind bude aplikácia, ktorá umožní navigáciu k najbližšej BTS pomocou kompasu na displeji. Užívateľ bude mať tiež možnosť navigácie na ním zvolené miesto. Základom aplikácie bude mapa, na ktorej budú zobrazené BTS. BTS, ktorá sa nachádza najbližšie k mobilnému zariadeniu bude zvýraznená. K dispozícii bude taktiež zobrazenie mapy pokrytia operátora a možnosť zobraziť výškové dáta na mape.

4.2 Porovnanie s podobnými aplikáciami

Táto kapitola sa zaoberá porovnaním aplikácií s podobnou funkcionalitou. Tieto aplikácie však neposkytujú riešenia nezávislé na dátovom pripojení. Pre porovnanie pozri tabuľku 1.

4.2.1 Open Signal

Aplikácia Open Signal nás dokáže navigovať k aktuálne obsluhujúcej BTS. K dispozícii máme mapu pokrytia, ktorá sa pri načítaní uloží do pamäte cache, takže je možné ju neskôr zobraziť bez pripojenia k internetu. To isté je možné s BTS, avšak tie sa načítajú podľa aktuálnej polohy. Preto je potrebné byť fyzicky na mieste, kde chceme BTS načítať. Pri strate signálu nás aplikácia naviguje na sever. Máme teda k dispozícii mapu pokrytia a BTS, ale aplikácia nám neposkytuje ďalšie údaje ako sú smer alebo vzdialenosť k najbližšej BTS.

4.2.2 Network Signal Info

Poskytuje informácie o mobilnej sieti, ale tiež o WiFi a systéme. Ako mapový podklad využíva Google Maps. Podporuje ako normálne, tak aj satelitné zobrazenie. Problém nastane, ak na mobilnom zariadení nemáme aktívne pripojenie k internetu, alebo ak stratíme signál mobilnej siete. V takomto prípade nie je možné mapu zobraziť. Navigácia k BTS nie je možná.

4.2.3 RF Signal Tracker

RF Signal Tracker slúži na monitorovanie RSSI, zobrazenie polohy obsluhujúcej BTS a zobrazenie teoretického pokrytia. Podobne ako Network Signal Info aj RF Signal Tracker poskytuje informácie o systéme a aktuálne pripojenej sieti. Máme možnosť vybrať si, z ktorej databázy aplikácia získava informácie o okolitých BTS a to: Google, OpenCellId

Názov aplikácie	Open Signal	Network Signal Info	RF Signal Tracker	SigFind
Mapová komponenta	Google Maps	Google Maps	Google Maps	Mapsforge
Navigácia k BTS	Iba k obsluhujúcej	Iba k obsluhujúcej	Iba k obsluhujúcej	K najbližšej
Navigácia na zadané miesto	Nie	Nie	Nie	Áno
Informácie o systéme a sieti	Základné info o sieti	Áno	Áno	Nie
Mapa pokrytia	Áno	Nie	Áno	Áno
Offline	Čiastočne	Nie	Čiastočne	Áno

Tabuľka 1: Porovnanie aplikácií

alebo lokálna DB. Pri zvolenej lokálnej DB, aplikácia zbiera údaje o pripojených BTS, ktoré ukladá do zariadenia. Aplikácia využíva mapy Google Maps s podporou satelitného zobrazenia. Bez pripojenia k internetu je možné využívat'len lokálnu DB, v ktorej nemusia byt'všetky dostupné BTS. Aplikácia nás naviguje k aktuálne obsluhujúcej BTS. V prípade straty signálu navigácia nie je aktívna.

4.3 Získavanie dát

Aplikácia pre svoj chod nepotrebuje internetové pripojenie. Užívateľ si musí stiahnut' súbor s mapou do zariadenia vopred. Ak užívateľ požaduje vykresľovanie topografickej vrstvy na mape, mal by si tiež stiahnut' potrebné súbory s SRTM dátami. Ostatné dáta potrebné pre chod aplikácie budú stiahnuté pri prvom spustení, alebo ak by boli zo zariadenia odstránené.

4.3.1 Informácie o BTS

Pri hľadaní informácií o BTS sa ukázal ako najlepší zdroj GSMweb.cz. Chvíľu sme uvažovali o DB OpenCellID, ktorá obsahuje údaje ako dosah BTS, alebo priemerná sila signálu [6]. Tieto údaje sú väčšinou nevyplnené a GPS súradnice BTS nie sú presné, ale získavané pomocou meraní. Pre lepšiu presnost' a kompletnost' sme sa rozhodli pre GSMweb.cz. GSMweb.cz poskytuje vytvorené zoznamy BTS v rôznych formátoch ako napríklad CSV, TXT, alebo HTML. Problém je, že v týchto zoznamoch nie sú GPS súradnice BTS (viac v 6.3.2).

Ak bude aplikácia nasadená a dostupná na Google Play, bolo by dobré získavat' dáta priamo od operátorov. Tak by sme mali dostupné všetky potrebné informácie o BTS a bolo by možné zobrazit' teoretické pokrytie vypočítané z týchto údajov.

4.3.2 Mapy pokrytia operátorov

Operátori majú väčšinou na webe k dispozícii mapy pokrytia. Mapa pokrytia zobrazuje teoretické pokrytie geodetického územia signálom. Pre ukážku sme použili mapu pokrytia od O2. Na získanie tejto mapy bolo treba sledovať požiadavky stránky a napísať skript, ktorý stiahne danú mapu pre určitú úroveň priblíženia.

4.4 Diagram tried

V diagrame tried (Obrázok 1 a Obrázok 2) sú zobrazené triedy aplikácie rozdelené do balíkov podľa ich funkcionality.

V hlavnom balíku a balíku *sigfind.filepicker* sa nachádzajú triedy, ktoré sa starajú o UI a interakciu s užívateľom. Trieda *FilePickerAdapter* poskytuje triede *FilePicker* informácie o tom, ako má byť vykreslený obsah v *GridView*.

Balík *sigfind.data* obsahuje triedy, ktoré reprezentujú objekty jednotlivých tabuliek DB, až na triedu *DatabaseHelper*, ktorá je potrebná na vytvorenie a prácu s DB.

V balíku *sigfind.filefilter* sú triedy implementujúce rozhranie *FileFilter*. Slúžia na filtrovanie súborov zobrazovaných triedou *FilePicker* podľa prípony, alebo podľa typu súboru.

V balíku *sigfind.util* je trieda *DatabaseUpdater*, ktorá slúži na aktualizovanie údajov o BTS v lokálnej databáze.

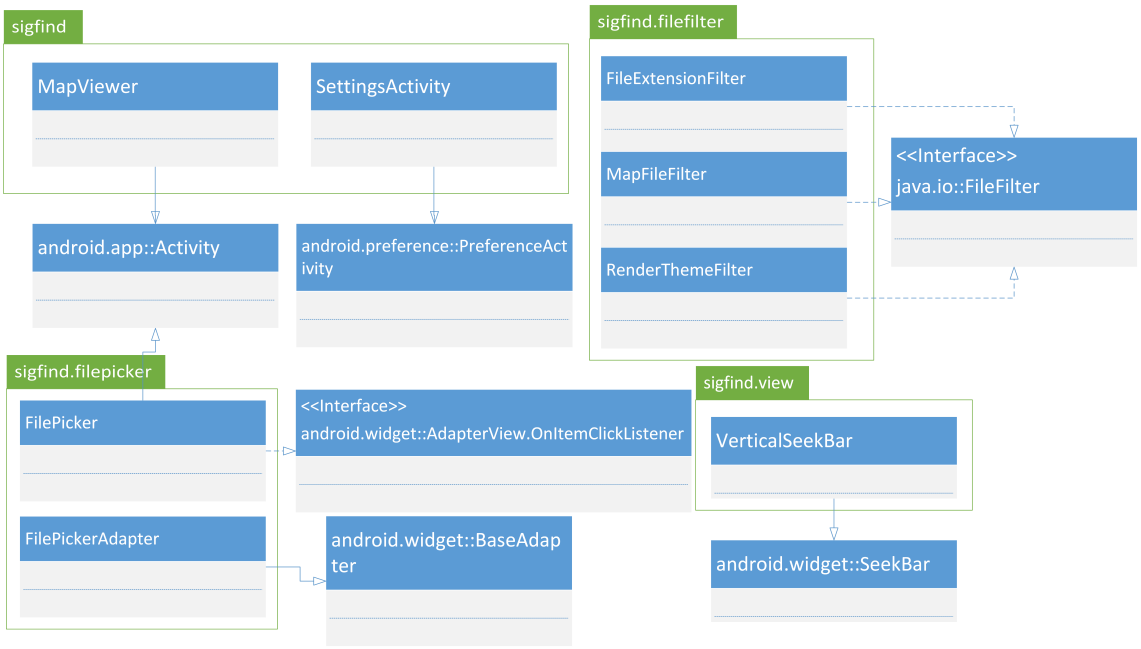
Ďalší balík je *sigfind.view*, v ktorom je trieda *VerticalSeekBar* predstavujúca vertikálnu verziu *android.widget.SeekBar*.

V poslednom balíku *sigfind.overlay* sú triedy reprezentujúce vrstvy zobrazované na mape ako je napríklad trieda *MyLocation*, ktorá vykresľuje na mapu aktuálnu polohu zariadenia, alebo *SignalOverlay*, ktorá vykresľuje mapu pokrytia operátora.

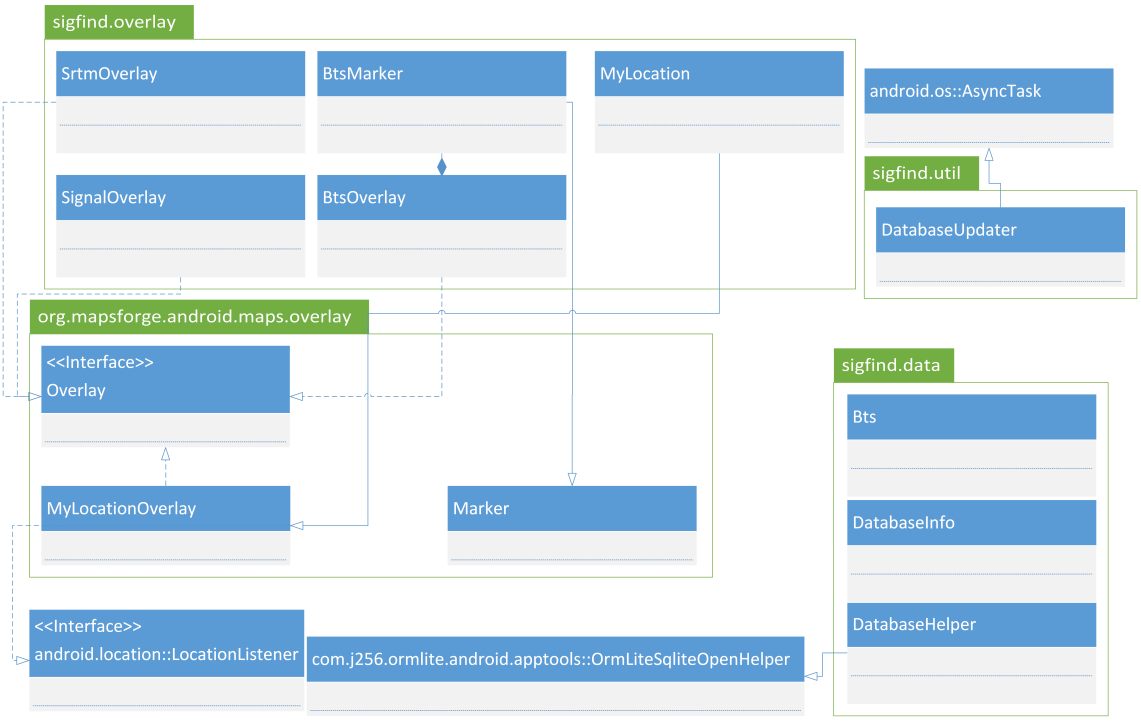
4.5 Diagramy aktivít

Na obrázku 3 je zachytený diagram aktivít popisujúci výber súboru (s mapou, témou, SRTM údajmi). Najskôr sa vypíšu všetky súbory v adresári a užívateľ si zvolí súbor ktorý chce vybrať, alebo zložku, v ktorej sa súbor nachádza. Pokiaľ je vybraný súbor zložka, vypíšu sa súbory v tejto zložke. Ak je to súbor, pokračuje sa na kontrolu súboru. Ak je typ súboru správny, nasleduje nastavenie tohoto súboru. V opačnom prípade sa zobrazí hláška a užívateľ musí vybrať iný súbor.

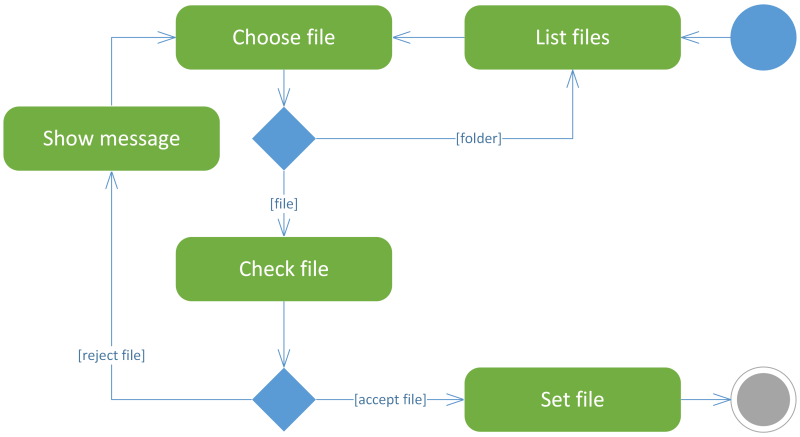
Na ďalšom diagrame aktivít (Obrázok 4) je zobrazená postupnosť aktivít pri aktualizovaní DB. Ako prvé sa overuje pripojenie k sieti. Pokiaľ nie je pripojenie aktívne, zobrazí sa hláška a užívateľ má možnosť pripojenie aktivovať. Ak ho užívateľ aktivuje, opäť sa skontroluje. Ak je aktívne, overí sa, či je potreba aktualizácie. Keď je DB aktuálna, aktualizácia sa nevykoná. V prípade potreby aktualizácie DB sa stiahnu dáta, ktoré sa následne pomocou parsera analyzujú, a vybrané dáta sa uložia do DB. Paralelne s týmito aktivitami sa užívateľovi zobrazí priebeh aktualizácie.



Obr. 1: Návrh tried UI



Obr. 2: Návrh dátovej vrstvy a vrstiev mapy



Obr. 3: Diagram aktivít – výber súboru

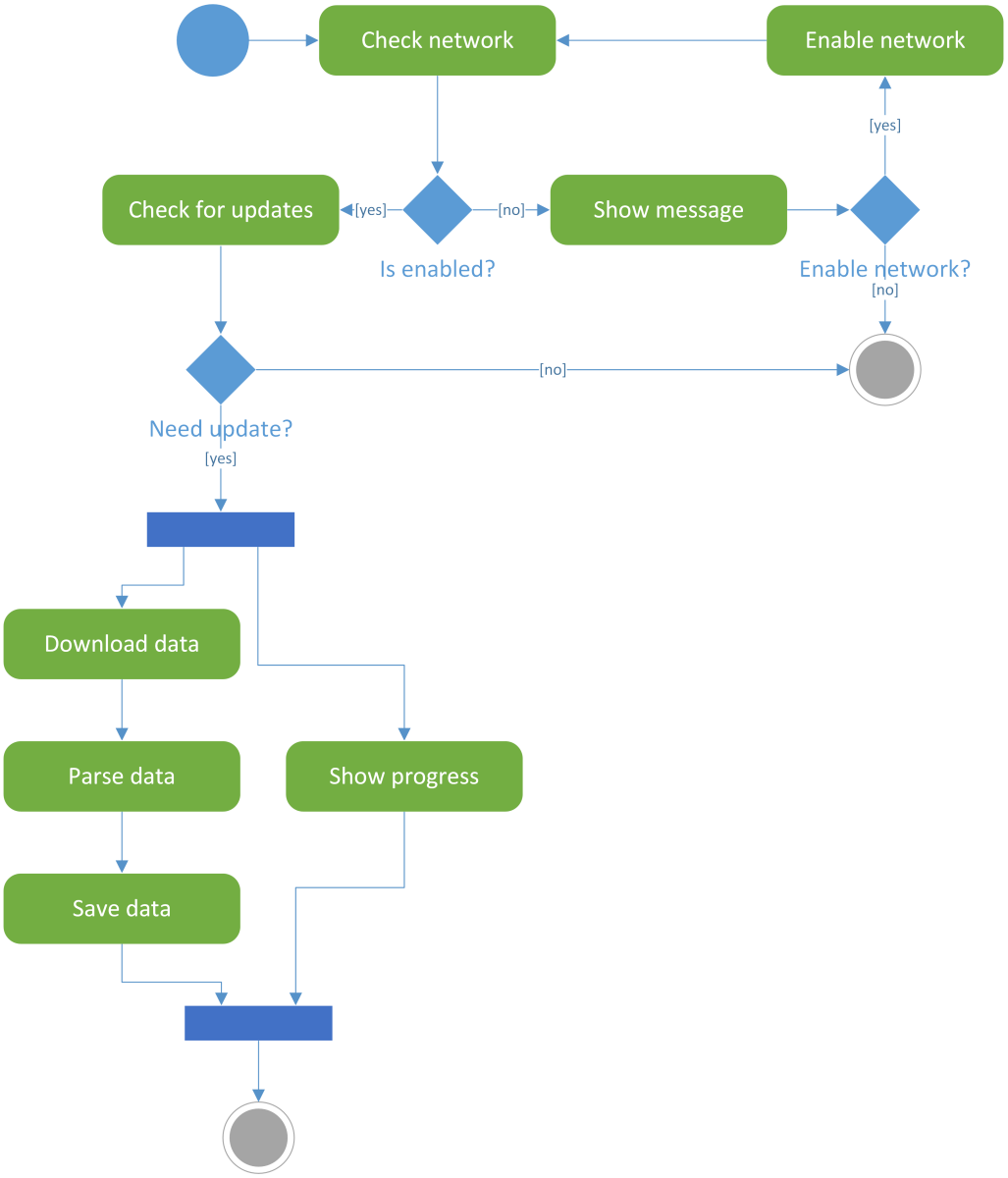
4.6 Sekvenčný diagram

Obrázok 5 predstavuje sekvenčný diagram pre aktualizáciu DB. Sekvencia správ začne, keď užívateľ zadá voľbu aktualizácie DB. Trieda *SettingsActivity* spustí aktualizáciu DB zavolaním metódy *execute()* triedy *DatabaseUpdater*. Tá potom z DB získa dátum jej vytvorenia (aktualizácie) a porovná ho s dnešným dátumom. Ak sa zhodujú, databáza je aktuálna. Ak sa nezhodujú, nasleduje získanie dát od triedy *Parser*. Následne sa tieto dáta prejdú a vytvoria sa objekty triedy *Bts*, ktoré sa uložia pomocou metódy *createOrUpdate(bts)* triedy *Dao*. Po dokončení slučky sa do DB vloží dátum poslednej aktualizácie a triede *SettingsActivity* sa pošle správa o jej dokončení. Tá ju následne zobrazí užívateľovi.

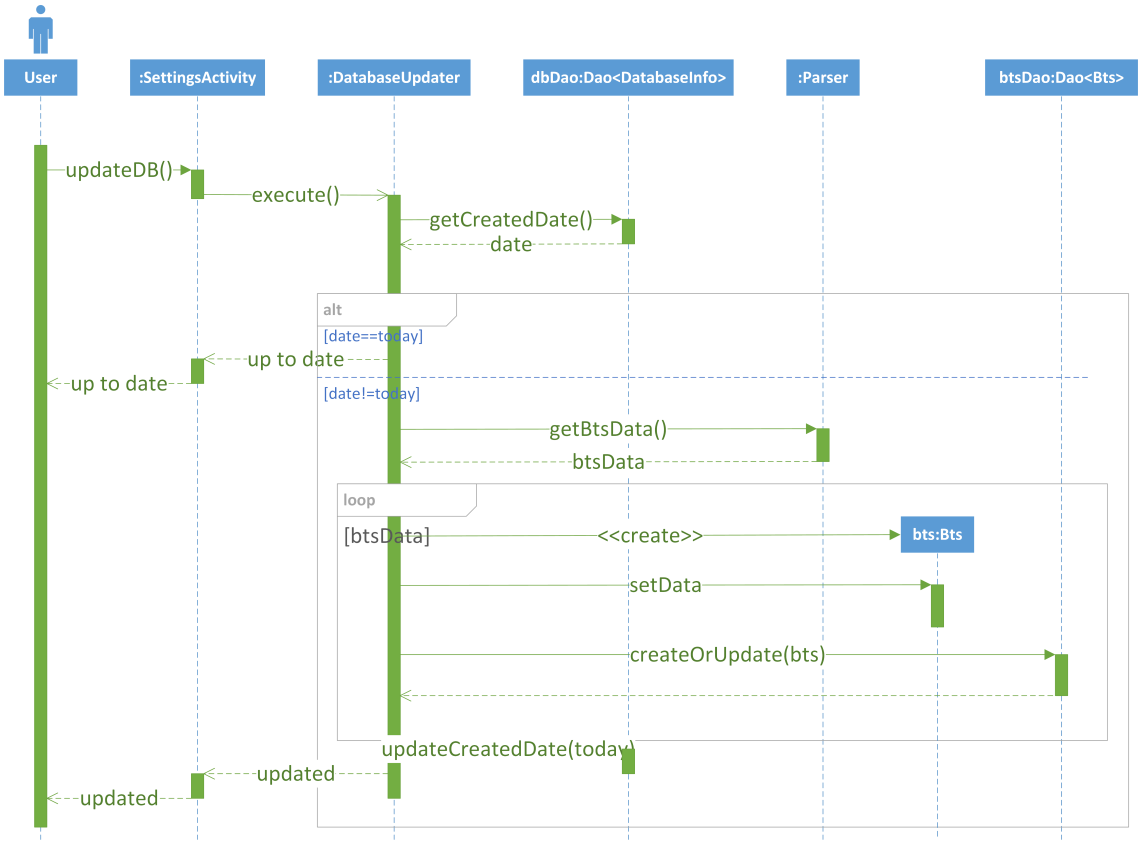
4.7 Dátový slovník

Bts	Typ	Kľúč	NULL	INDEX	IO
cellId	INTEGER	áno	nie	áno	-
city	TEXT	nie	nie	nie	-
mnc	INTEGER	nie	nie	nie	-
mcc	INTEGER	nie	nie	nie	-
lac	INTEGER	nie	nie	nie	-
range	REAL	nie	áno	nie	-
latitude	REAL	nie	nie	áno	-
longitude	REAL	nie	nie	áno	-
DatabaseInfo					
id	INTEGER	áno	nie	áno	-
created	TEXT	nie	nie	nie	-

Tabuľka 2: Dátový slovník



Obr. 4: Diagram aktivít – aktualizácia databázy

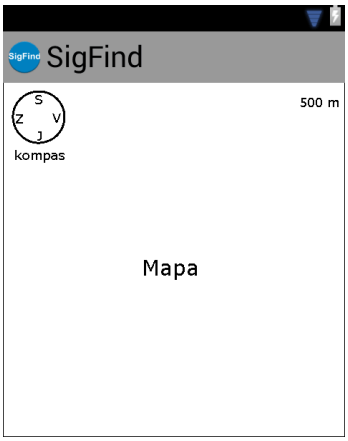


Obr. 5: Sekvenčný diagram – aktualizácia databázy

5 Návrh

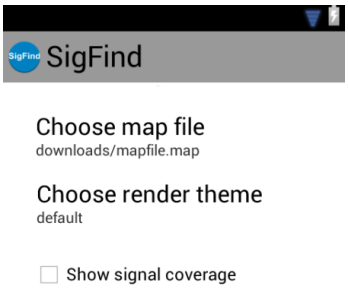
5.1 Užívateľské rozhranie

Hlavným prvkom užívateľského rozhrania je mapa, preto je UI navrhnuté v minimalistickom štýle (Obrázok 6). Okrem mapy je k dispozícii kompas, ktorý slúži ako navigácia k najbližšej BTS, alebo k bodu zvolenému užívateľom.



Obr. 6: Zobrazenie mapy

Pri návrhu UI nastavení aplikácie (Obrázok 7) bol braný ohľad na konzistenciu s inými Android aplikáciami, ako aj so systémovými nastaveniami.



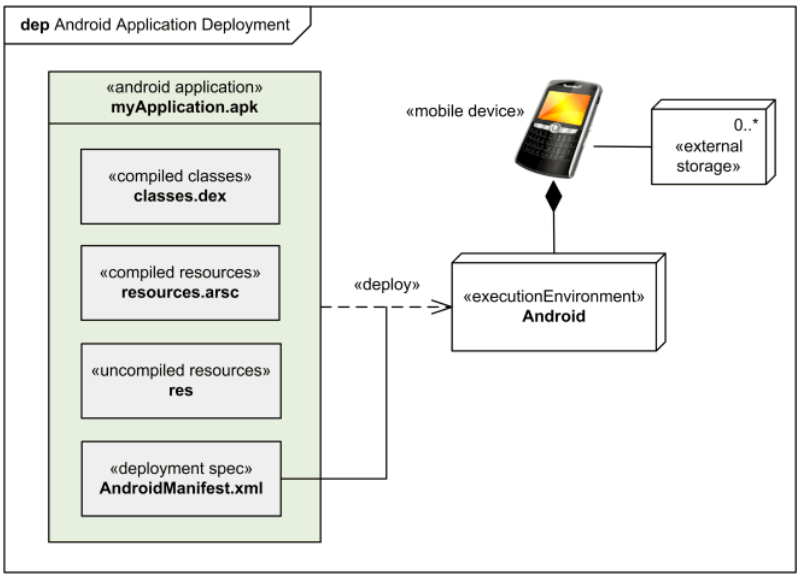
Obr. 7: Nastavenia aplikácie

Pri výbere súboru s mapou, alebo XML témy (Obrázok 8) sa zobrazujú priečinky a súbory zodpovedajúce danej prípone súboru, poprípade typu súboru.

Obr. 8: Výber súboru

5.2 Diagram nasadenia

Nástroje Android SDK kompilujú a balia kód spolu s potrebnými dátami a zdrojovými súbormi do archívneho súboru Android aplikácie majúceho príponu .apk. Súbor .apk reprezentuje jednu Android aplikáciu, ktorá má byť zavedená do mobilného zariadenia so systémom Android (Obrázok 9).



Obr. 9: Príklad zavedenia aplikácie – prevzaté z [7]

6 Implementácia

6.1 Použité knižnice

Pri výbere mapovej komponenty som najskôr vyskúšal použitie Google Maps a hľadal možnosti offline použitia OpenStreetMap. Po odskúšaní Google Maps som skúsil prácu s osmdroid, ale nakoniec som sa rozhodol pre použitie mapsforge kvôli jednoduchosti pri použití offline máp.

Manuálne získavanie dát z HTML kódu býva niekedy náročné a pomalé. Preto som sa rozhodol použit' knižnicu, ktorá parsovanie HTML automatizuje. Takých knižníc však existuje veľa. Najviac ma zaujala knižnica jsoup, ktorá dokáže stiahnuť dokument a následne v ňom vyhľadat' potrebné informácie za použitia napr. aj CSS selektorov. Taktiež má výbornú dokumentáciu.

Posledná použitá knižnica je OrmLite. Táto knižnica poskytuje funkcionalitu väčších ORM, pričom si zachováva jednoduchost' a malú veľkost', čo je v mobilných aplikáciách veľmi dôležité. Práca s DAO a *QueryBuilderom* je jednoduchšia ako použitie kurzorov (*Cursor*) a *SQLiteQueryBuilder* v androide.

6.1.1 Mapsforge

Knižnica Mapsforge umožňuje vykresľovanie vektorových máp za behu aplikácie. Mapy sú generované z mapových súborov vo formáte .map, ktoré sa dajú vytvorit' pomocou nástroja Map-Writer. Map-Writer je dostupný ako Osmosis plugin [8].

Štýly máp sa dajú prispôsobit' pomocou konfiguračných XML súborov. Tento XML súbor obsahuje pravidlá a renderovacie inštrukcie. (viď Výpis 4) Mapsforge má zabudovanú defaultnú renderovaciu tému a v budúcnosti je v pláne pridávanie ďalších tém. Mapsforge takisto podporuje použitie externých tém. Externú tému je možné nastavit' použitím metódy *MapView.setRenderTheme(File)* [9].

```
<?xml version="1.0" encoding="UTF-8"?>
<rendertheme xmlns="http://mapsforge.org/renderTheme" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance"
  xsi:schemaLocation="http://mapsforge.org/renderTheme renderTheme.xsd" version="1">
  <!-- matches all ways with a "highway=trunk" or a "highway=motorway" tag -->
    <rule e="way" k="highway" v="trunk|motorway">
      <line stroke="#FF9900" stroke-width="2.5" />
    </rule>
    <!-- matches all closed ways (first node equals last node) with an "amenity=..." tag -->
    <rule e="way" k="amenity" v="*" closed="yes">
      <area fill ="#DDEECC" stroke="#006699" stroke-width="0.3" />
    </rule>
    <!-- matches all nodes with a "tourism=hotel" tag on zoom level 16 and above -->
    <rule e="node" k="tourism" v="hotel" zoom-min="16">
      <symbol src="file:/path/to/symbol/icon/hotel.png" />
      <caption k="name" font-style="bold" font-size="10" fill="#4040ff" />
    </rule>
</rendertheme>
```

Výpis 4: Príklad renderovacej témy mapsforge [9]

Použitie knižnice mapsforge v Android aplikácii je jednoduchšie ako použitie Google Maps vzľadom na to, že nie je treba nastavovať kľúč API, ani ďalšie oprávnenia (viď Výpis 5). Jediné oprávnenie potrebné pre pužitie mapsforge je *WRITE_EXTERNAL_STORAGE*, ktoré je potrebné na ukladanie generovaných obrázkov mapy na externé úložisko. V aplikácii je použitá aktuálna verzia knižnice mapsforge 0.3.1.

```
import android.os.Bundle;
import java.io.File;
import org.mapsforge.android.maps.MapActivity;
import org.mapsforge.android.maps.MapView;

public class MApViewer extends MapActivity {
    private MapView mMapView;
    /**
     * Set properties of MapView
     */
    private void setMapViewProperties() {
        mMapView.setClickable(true);
        final MapZoomControls zc = mMapView.getMapZoomControls();
        zc.setShowMapZoomControls(true);
        zc.setZoomLevelMin((byte) 7);
        mMapView.getMapMover()
                .setMoveSpeedFactor(0.5f);
        mMapView.getFileSystemTileCache()
                .setCapacity(FILE_SYSTEM_CACHE_SIZE);
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d("state", "onCreate");
        setContentView(R.layout.activity_map_viewer);
        mMapView = (MapView) findViewById(R.id.fullscreen_content);
        setMapViewProperties();
    }
}
```

Výpis 5: Príklad použitia knižnice mapsforge

6.1.2 OrmLite

Object Relational Mapping Lite poskytuje jednoduchú, odľahčenú funkcionalitu pre ukladanie Java objektov do SQL databázy. Podporuje Sqlite s natívnym volaním funkcií rozhrania API database v OS Android [10]. Umožňuje použitie Java anotácií v triedach, ktoré majú byť uložené do DB. Poskytuje *QueryBuilder* pre jednoduché vytváranie dotazov a abstraktné triedy DAO.

6.1.3 Jsoup

Jsoup je knižnica pre prácu s HTML. Poskytuje API vhodné na získavanie dát z HTML. Umožňuje používanie syntaxe podobnej jQuery, manipuláciu s HTML elementmi, atri-

bútmi a textom [11].

6.2 Implementácia overlays

Pri vytváraní aplikácie som sa držal analýzy a pôvodného návrhu. Zo začiatku som používal knižnicu mapsforge vo verzii 0.3.0, neskôr som prešiel na verziu 0.3.1. Preto sa návrh trochu pozmenil. Nebolo treba implementovať *LocationListener*, pretože v novšej verzii už je implementované *MyLocationOverlay*, ktoré sa stará o zobrazenie aktuálnej polohy. Túto triedu som však musel rozšíriť, kvôli výpočtom, ktoré sa musia vykonať pri zmene polohy (zobrazenie BTS v okolí, výpočet vzdialenosti k cieľu, výpočet azimutu).

6.2.1 Výpočet vzdialenosti k cieľu

Pri každej zmene polohy sa volá metóda *calculateDistanceAndBearing* v aktivite *MapViewer*. V nej sa vypočíta vzdialenosť a azimut k miestu určnému užívateľom, alebo sa zavolá metóda *calculateDistanceAndBearing* triedy *BtsOverlay*, ktorá vypočíta vzdialenosť a azimut k najbližšej BTS (viď Výpis 6).

```
public synchronized void calculateDistanceAndBearing(Location currentLocation) {
    distance = 50000000;
    synchronized (this.overlayItems) {
        int numberOfOverlayItems = this.overlayItems.size();
        Location l = new Location("Closest bts");
        for (int i = 0; i < numberOfOverlayItems; ++i) {
            BtsMarker btsMarker = this.overlayItems.get(i);
            l.setLatitude(btsMarker.getGeoPoint().latitude);
            l.setLongitude(btsMarker.getGeoPoint().longitude);
            float tempDistance = currentLocation.distanceTo(l);
            if (tempDistance < distance) {
                btsMarker.setActive(true);
                if (activeId >= 0 && activeId < numberOfOverlayItems)
                    this.overlayItems.get(activeId).setActive(false);
                distance = Math.round(tempDistance);
                bearing = currentLocation.bearingTo(l);
                activeId = i;
            }
        }
    }
    onCalculationFinishedListener.onCalculationFinish(bearing, distance);
}
```

Výpis 6: Výpočet vzdialenosti k najbližšej BTS

6.2.2 Vykreslenie mapy pokrytia

Mapa pokrytia je zobrazená ako *Overlay*, na ktorom sa vykresľujú dlaždice (bitmapy). Knižnica mapsforge neposkytuje špeciálne typy *Overlay* s možnosťou vykreslenia dlaždíc. Preto musíme implementovať triedu *Overlay* a v metóde *draw(..)* vykresliť dlaždice manuálne. Pomocou metód triedy *MercatorProjection* prevedieme súradnice zobrazenej

časti mapy na pixely a tie následne na súradnice požadovaných dlaždíc pre určitú úroveň priblíženia. Tieto súradnice sa prechádzajú v cykle a dlaždice s danými súradnicami sa vykresľujú na *Canvas* 7. Pre rýchlejší prístup k obrázkom dlaždíc je použitá medzipamäť a to použitím triedy *LRUCache*. Veľkosť pamäte je nastavená na $1/8$ celkovej veľkosti pamäte dostupnej aplikácii.

```
for (long tileY = tileTop; tileY <= tileBottom; ++tileY) {
    for (long tileX = tileLeft ; tileX <= tileRight; ++tileX) {
        Tile tile ;
        float left ;
        float top;
        Bitmap bitmap;

        // calculate pixel position of bitmap
        if (zoomLevel > MAX_ZOOM) {
            int Y = (int) MercatorProjection.latitudeToTileY(MercatorProjection.tileYToLatitude( tileY
                , MAX_ZOOM), zoomLevel);
            int X = (int) MercatorProjection.longitudeToTileX(MercatorProjection.tileXToLongitude(
                tileX, MAX_ZOOM), zoomLevel);
            tile = new Tile(X, Y, zoomLevel);
        } else {
            tile = new Tile(tileX , tileY , zoomLevel);
        }
        left = (float) ( tile .getPixelX() - pixelLeft);
        top = (float) ( tile .getPixelY() - pixelTop);

        // if zoomLevel > MAX_ZOOM, show tile from MAX_ZOOM
        String key = (zoomLevel > MAX_ZOOM ? MAX_ZOOM : zoomLevel) + "/" + tileX + "/" + tileY
            + ".png";

        if (cache.containsKey(key)) {
            bitmap = cache.get(key);
        } else {
            bitmap = BitmapFactory.decodeFile(TILE_FOLDER + key);
            if (bitmap == null)
                continue;
            cache.put(key, bitmap);
        }

        if (bitmap != null) {
            if (zoomLevel > MAX_ZOOM) {
                int multiplier = (int) Math.pow(2, zoomLevel - MAX_ZOOM);
                int dstHeightWidth = multiplier * Tile.TILE_SIZE;
                bitmap = Bitmap.createScaledBitmap(bitmap, dstHeightWidth, dstHeightWidth, false);
            }
            canvas.drawBitmap(bitmap, left, top, paint);
        }
    }
}
```

Výpis 7: Vykresľovanie dlaždíc mapy pokrytia

6.2.3 Vykreslenie topografickej vrstvy

Pri vykresľovaní topografickej vrstvy je použitý lineárny prechod medzi dvoma farbami, a to čiernou a bielou s nastavením priehľadnosti. Jedná sa o vykresľovanie tieňovania do mapy. Výšky sa mapujú na tieto farby, pričom biela predstavuje maximálnu a čierna minimálnu výšku (viď Výpis 8). Vykresľovanie topografickej vrstvy nemusí mať vždy význam, preto je možné zobrazovanie vrstvy vypnúť v nastaveniach aplikácie.

Dáta SRTM nemusia byť zobrazené iba ako tieňovanie reliéfu, ale tiež ako vrstevnice. Toto zobrazenie mapy je v alpikácii možné použitím máp s vrstevnicami a k nim dostupnej témy, ktorá určuje pravidlá ich vykresľovania.

```
private int getColor(int height) {
    // white 255
    // black 0
    if (height < hmin)
        height = hmin;
    if (height > hmax)
        height = hmax;
    float k = height / 2000f;

    return (int) (k * 255 + (1 - k) * 0 + 0.5);
}
```

Výpis 8: Mapovanie výšok na farby

6.3 Problémy a ich riešenie

6.3.1 Problém s pamäťou

Každá Android aplikácia má pridelené určité množstvo pamäte, ktoré nesmie prekročiť. V opačnom prípade systém vyvolá výnimku *OutOfMemory*. Táto pamäť sa nazýva halda (heap) a sú v nej uložené dynamicky alokované premenné. Množstvo pridelenej pamäte záleží od zariadenia, na ktorom aplikácia beží.

Pri vykresľovaní mapy pokrytia ako *Overlay* nastal problém s nedostatkom pamäte. K dispozícii máme mapu pokrytia pre najvyššiu úroveň priblíženia 12. Ak je priblíženie detailnejšie, musí sa mapa generovať z úrovne 12 jej zväčšením. Zväčšený objekt Bitmap je napríklad pre úroveň priblíženia 13 štyrikrát väčší oproti úrovni 12. Pokiaľ sa má mapa pokrytia vykresliť pre úroveň 15, na niektorých zariadeniach s malou pamäťou môže dôjsť k pretečeniu.

Riešením problému je vykresľovanie mapy pokrytia do úrovne priblíženia 14. Pri detailnejšom priblížení sa mapa pokrytia nevykreslí.

6.3.2 Problém so získavaním dát

Potrebné údaje o BTS sú získavané z GSMweb.cz. V poskytovaných zoznamoch BTS však nie sú dostupné informácie o lokalizácii jednotlivých BTS. Tieto údaje sú pre aplikáciu kritické.

Vo vyhľadávaní na stránke GSMweb.cz sú dostupné všetky potrebné informácie o BTS spolu s odkazmi na mapu. V týchto odkazoch sa nachádzajú súradnice GPS. Pre ich získanie je použitá knižnica Jsoup (viď 6.1.3).

6.3.3 Deadlock v overlays

Rátanie vzdialenosti k BTS a odchýlenia kompasu je závislé na aktuálnej polohe zariadenia. Pri každej zmene polohy je potrebné prerátať tieto hodnoty. V pôvodnom riešení trieda *MyLocation* (viď Obr. 3.1) pristupovala k triede *BtsOverlay*. Metódy v týchto triedach sú synchronizované, aby k nim nebolo možné pristupovať naraz z viacerých vlákien. Problém nastal pri volaní metódy *redrawOverlays()*. Na zozname, ktorý vracia metóda *getOverlays()*, sa nastavil zámok a trieda *MyLocation* nemohla pristupovať k triede *BtsOverlay*. Aplikácia teda na tomto kóde zastavila a čakala, kedy sa uvoľní zámok na zozname overlays, aby mohla trieda *MyLocation* dokončiť svoju prácu. Tento stav sa nazýva deadlock. Zvláštne na tejto situácii bolo, že systém na ňu vôbec nereagoval, hoci aplikácia bola nečinná a nebolo možné ju ovládať. V takejto situácii Android zvyčajne zobrazí správu "Application Not Responding" (ANR).

Pre odstránenie tohto problému je v triede *MapViewer* metóda, ktorá v prípade potreby zavolá metódu na výpočet vzdialenosti k najbližšej BTS, alebo polohy určenej používateľom. Po dokončení výpočtu sa aktualizujú hodnoty v UI. Týmto spôsobom je UI vlákno odľahčené od výpočtov a zároveň nemôže nastať opisovaná situácia s deadlock-om.

7 Testovanie

Aby sme mohli aplikáciu testovať, je potrebné určiť aktuálnu polohu mobilného zariadenia. Toto je možné prostredníctvom GPS modulu zariadenia. Bez toho sa v oblasti bez pokrytia signálom aplikácia nedá správne používať.

Po určení polohy zariadenia aplikácia vypočíta vzdialenosť (m) k najbližšej BTS, pričom sa táto vzdialenosť zobrazí v pravom hornom rohu displeja zariadenia (Obrázok 10(a)). Pri dlhšom dotyku displeja sa na mape zobrazí ikona v mieste označenia, určujúca nový cieľ. Vzdialenosť k novému cieľu sa automaticky prepočíta. Aplikácia má zároveň slúžiť ako navigácia. Smer k cieľovému bodu určuje kompas umiestnený v ľavom hornom rohu displeja. V prípade, že aplikácia nemá k dispozícii cieľový bod, kompas ukazuje na sever a vzdialenosť je neznáma ("unknown").

Aplikácia poskytuje možnosť zobrazenia mapy pokrytia operátora (Obrázok 10(b)). Táto možnosť je pri prvom spustení predvolená v nastaveniach, kde je možné ju aj vypať. V ľavej časti obrazovky sa nachádza posuvník, ktorý umožňuje zmeniť priehľadnosť tejto mapy.

Pre zobrazenie výškových dát na mape, je potrebné vybrať v nastaveniach súbor s SRTM dátamim a uistiť sa, že vykreľovanie topografickej vrstvy je zapnuté. (Pre porovnanie mapy s vykreslenou topografickou vrstvou a mapy bez nej pozri obrázok 10(c) a obrázok 10(d).) Okrem toho si môžeme vybrať súbor s mapou, ktorý obsahuje vrstevnice a k nemu príslušnú tému pre ich správne vykreslenie (Obrázok 10(e)). Aplikácia však používa verziu 0.3.1 knižnice mapsforge a je nutné používať témy vo verzii 2.

Aplikácia bola testovaná prevažne na zariadení HTC Evo 3D, kde všetko fungovalo podľa očakávaní. Okrem fyzického zariadenia bola testovaná v niekoľkých virtuálnych zariadeniach behom vývoja. Tým bolo možné vidieť, ako sa aplikácia správa na novšej, poprípade staršej verzii systému Android. Konkrétne bola otestovaná na zariadeniach s verziou Android 2.3.3 a Android 4.2. Keďže zariadenia s verziou systému Android 4.2 neposkytujú fyzické tlačidlo pre zobrazenie menu, je v tomto prípade zobrazená ikona v action bare, ktorá tlačidlo menu nahrádza.

8 Záver

Cieľom práce bolo zistiť možnosti lokalizácie pre platformu Android, dostunposť údajov o BTS z verejných zdrojov a porovnanie existujúcich aplikácií s rovnakým, alebo podobným zameraním. Ciele predkladanej bakalárskej práce boli splnené.

Pri implementácii aplikácie sa vyskytli problémy, ktoré by bránili jej plnohodnotnému využitiu. Hlavným problémom bolo získavanie dát o BTS. Druhotným problémom bol nedostatok pamäte pri vykresľovaní vrstvy s pokrytím signálu.

Výsledná aplikácia bola testovaná prevažne na mobilnom zariadení HTC Evo 3D s verziou systému Android 4.0.3, ale aj na virtuálnych zariadeniach. Tak sme mohli porovnať, ako funguje v jednotlivých verziách systému Android.

Konečná verzia aplikácie je charakteristická tým, že užívateľom poskytuje možnosť offline navedenia do oblasti so signálom.

V budúcnosti sa môžu vyriešiť už spomínané problémy. V ďalšej verzii môžeme pridať upozornenie na zachytenie, prípadne stratu signálu. Ďalej by bolo vhodné vypočítavať teoretické pokrytie signálom z údajov o BTS a výškových dát, vylepšenie vykresľovania výškových dát. Možné je aj pridanie témy so zvýraznením obytných budov, ktorá by nahradila predvolenú tému. Ďalšia verzia by mala tiež používať knižnicu mapsforge vo verzii 0.4, ktorá prináša vylepšenia vo vykresľovaní vrstiev.

9 Literatúra

[1] Burnette, Ed, *Hello, Android: Introducing Google's Mobile Development Platform*, Pragmatic Bookshelf, 2008. 978-1-93435-617-3.

[2] Meier, Reto, *Professional Android 2 Application Development*, Wrox Press, 2010. 0-47056-552-0.

[3] The AndroidManifest.xml File, *Android Developers*, [Online] [Dátum: 20. 4. 2013], http://developer.android.com/guide/topics/manifest/manifest-intro.html.

[4] Application fundamentals, *Android Developers*, [Online] [Dátum: 20. 4. 2013], http://developer.android.com/guide/components/fundamentals.html.

[5] AsyncTask, *Android Developers*, [Online] [Dátum: 20. 4. 2013], http://developer.android.com/reference/android/os/AsyncTask.html.

[6] ENAiKOON, *OpenCellIDs*, [Online] [Dátum: 20. 4. 2013], http://www.opencellids.org/fileadmin/opencellid/explanation_of_database.txt.

[7] Application development for Android, *UML Diagrams*, [Online] [Dátum: 28. 4. 2013], http://www.uml-diagrams.org/examples/android-uml-examples.html.

[8] GettingStartedMapWriter, *Mapsforge*, [Online] [Dátum: 20. 4. 2013], http://code.google.com/p/mapsforge/wiki/GettingStartedMapWriter.

[9] RenderThemeAPI - Description of the render-theme API from the mapsforge-map library, *Mapsforge*, [Online] [Dátum: 3. 5. 2013], http://code.google.com/p/mapsforge/wiki/RenderThemeAPI.

[10] OrmLite - Lightweight Object Relational Mapping (ORM) Java Package, *OrmLite*, [Online] [Dátum: 20. 4. 2013], http://ormlite.com/.

[11] jsoup Java HTML Parser, with best of DOM, CSS, and jquery, *jsoup Java HTML parser*, [Online] [Dátum: 20. 4. 2013], http://jsoup.org/.

[12] Google Maps Android API v2, *Google Developers*, [Online] [Dátum: 1. 5. 2013], https://developers.google.com/maps/documentation/android/.

[13] Activities, *Android Developers*, [Online] [Dátum: 1. 5. 2013], http://developer.android.com/guide/components/activities.html.

[14] Rodriguez, E., C.S. Morris, J.E. Belz, E.C. Chapin, J.M. Martin, W. Daffer, S. Hensley, *An assessment of the SRTM topographic products*, Technical Report JPL D-31639, Jet Propulsion Laboratory, Pasadena, California, 2005, [Online] [Dátum: 2. 5. 2013], http://www2.jpl.nasa.gov/srtm/SRTM_D31639.pdf.

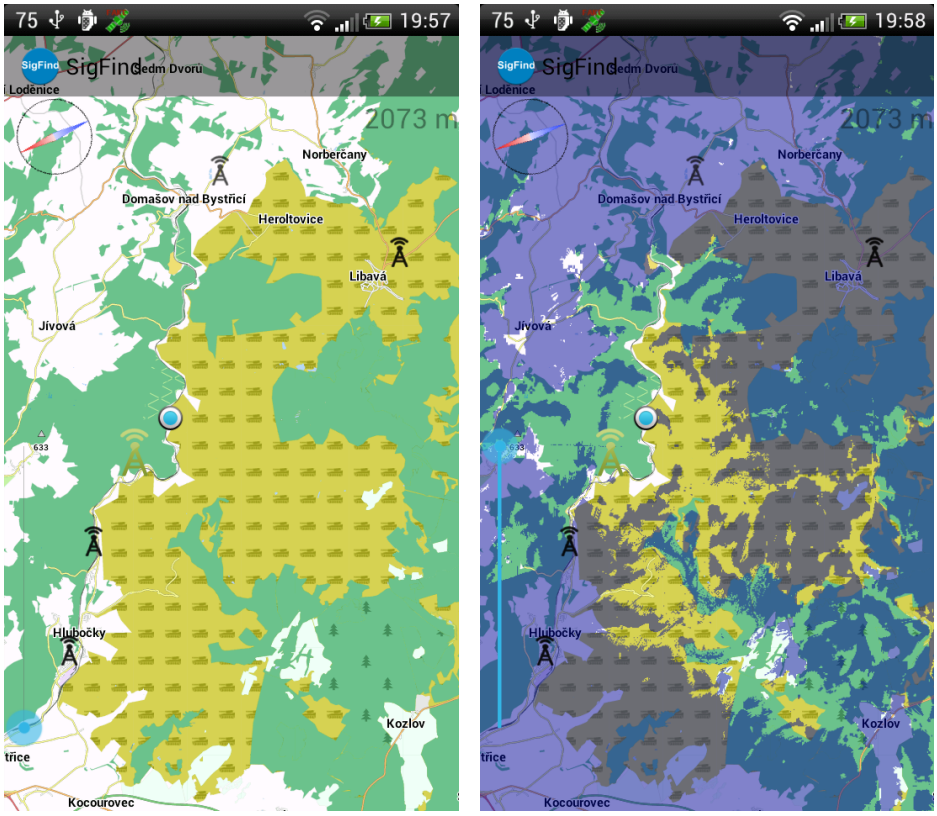
[15] Mission Overview, *Shuttle Radar Topography Mission*, [Online] [Dátum: 2. 5. 2013], http://www2.jpl.nasa.gov/srtm/missionoverview.html.

[16] What are these, [Online] [Dátum: 2. 5. 2013], http://dds.cr.usgs.gov/srtm/What_are_these.pdf.

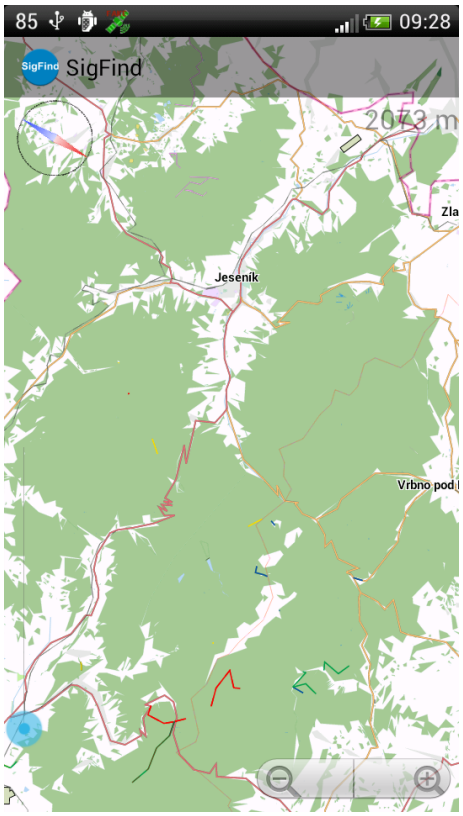
[17] SRTM Data Processing Methodology, *CGIAR-CSI SRTM 90m DEM Digital Elevation Database*, [Online] [Dátum: 6. 5. 2013], http://srtm.csi.cgiar.org/SRTMdataProcessingMethodology.asp.

[18] SRTM Topography, [Online] [Dátum: 2. 5. 2013], http://dds.cr.usgs.gov/srtm/version2_1/Documentation/SRTM_Topo.pdf.

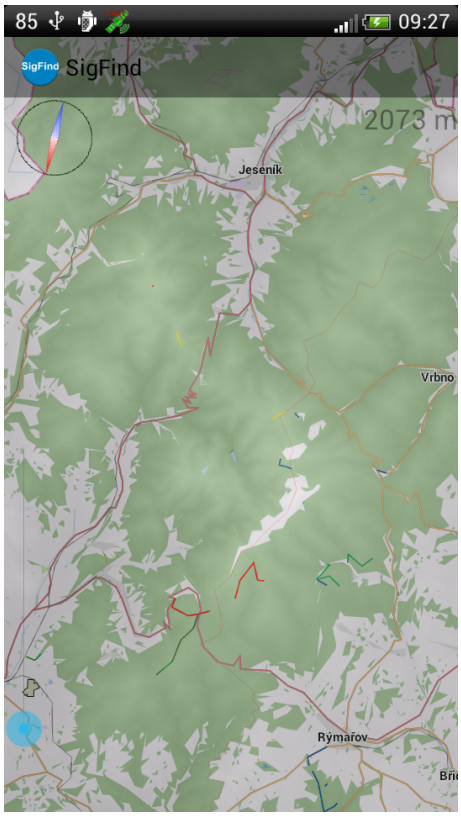
A Ukážky aplikácie



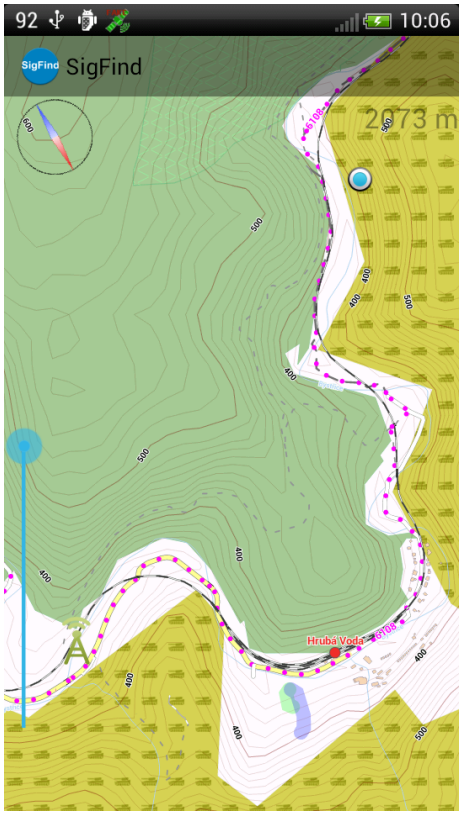
(a) Navigácia k najbližšej BTS (b) Zobrazenie mapy pokrytia



(c) Zobrazenie mapy bez topografickej vrstvy



(d) Zobrazenie mapy s topografickou vrstvou



(e) Ukážka mapy s vrstevnicami

B Obsah elektronickej prílohy

- **sigfind.apk** – inštalačný balíček aplikácie
- **hgt** – priečinok s výškovými dátami
- **czech_republic_gccz.map** – mapa Českej republiky
- **osmarender_gccz** – téma pre vykresľovanie vrstevníc
- **bp** – priečinok s textom bakalárskej práce
- **Súbor SigFind.zip**
  - libs – použité knižnice
  - res – súbory Android resources (animácie, ikony, layouty, hodnoty)
  - src/cz/vsb/sigfind – zdrojové kódy aplikácie
    - data – triedy pre prácu s DB
      - Bts.java
      - DatabaseInfo.java
      - DatabaseHelper.java
    - filefilter – triedy pre filtrovanie súborov
      - FileExtensionFilter.java
      - MapFileFilter.java
      - RenderThemeFilter.java
    - filepicker – implementácia prehliadača súborov
      - FilePicker.java
      - FilePickerAdapter.java
    - overlay – triedy implementujúce *Overlay*, alebo *Marker*
      - BtsMarker.java
      - BtsOverlay.java
      - CalculationFinishedListener.java
      - MyLocation.java
      - SignalOverlay.java
      - SrtmOverlay.java
    - util – nástroje
      - DatabaseUpdater.java
    - view – triedy rozširujúce *View*
      - VerticalSeekBar.java
    - AsyncTaskCompleteListener.java
    - MapViewer.java – hlavná aktivita
    - SettingsActivity.java – aktivita s nastaveniami aplikácie

- AndroidManifest.xml – manifest aplikácie
- ic_launcher_web.png – ikona aplikácie, zobrazovaná na webe (Google Play)